

Performance Evaluation of Vision-Based Algorithms for MAVs

T. Holzmann¹, R. Prettenthaler¹, J. Pestana^{1,3}, D. Muschick^{2*},
G. Graber¹, C. Mostegel¹, F. Fraundorfer¹ and H. Bischof¹

Abstract. An important focus of current research in the field of Micro Aerial Vehicles (MAVs) is to increase the safety of their operation in general unstructured environments. Especially indoors, where GPS cannot be used for localization, reliable algorithms for localization and mapping of the environment are necessary in order to keep an MAV airborne safely. In this paper, we compare vision-based real-time capable methods for localization and mapping and point out their strengths and weaknesses. Additionally, we describe algorithms for state estimation, control and navigation, which use the localization and mapping results of our vision-based algorithms as input.

1 Introduction

In the last years, much effort was put into research in the field of (semi-) autonomous Micro Aerial Vehicles (MAVs). Even though algorithms for autonomous control and navigation exist, most MAVs can work autonomously only in constrained environments (e.g., by using GPS, which is not available indoors). As the usage of MAVs in challenging environments for inspection and surveillance is a hot industrial issue, it is covered by the European Robotics Challenges ⁴(EuRoC). They were announced in order to stimulate research and development in robotics and support the transfer between academia and industry. We are currently participating in one of the challenges, which is targeted at Plant Servicing and Inspection using MAVs. In this paper, we evaluate algorithms for vision-based localization and reconstruction, state estimation, control and navigation using the simulation environment of the challenge.

For state estimation and control of the MAV, an accurate localization estimate is necessary. Several approaches exist to do this using visual sensors. For localization and reconstruction in real-time, a widely used system is PTAM [9]. PTAM uses a single camera as input and computes the pose of the camera and a map of the environment simultaneously by using sparse feature points. A different approach is proposed by Engel et al. [2]: Instead of using sparse image features, they use the intensity values of most of the pixels of the image directly to align the image to 3D points accordingly and estimate the pose of the camera. As the extraction and matching of features usually takes a lot of time, using the image intensities directly speeds up the localization process. However, as both of these approaches are using just a single camera, the scale of the reconstruction and localization cannot be determined and they may have problems with certain movements (e.g., pure rotations). In contrast, Geiger et al. [6] use stereo images as input and match sparse image features in order to get the relative

*The first four authors contributed equally to this work.

¹Institute for Computer Graphics and Vision and ²Institute of Automation and Control, Graz University of Technology; and ³CVG, Centro de Automática y Robótica (CAR, CSIC-UPM) {holzmann, pestana, mostegel, graber, fraundorfer, bischof}@icg.tugraz.at {daniel.muschick, rudolf.prettenthaler}@tugraz.at

⁴<http://www.euroc-project.eu/>

pose estimate. Having the depth data from the stereo image pair, it is possible to determine the correct scale and to handle pure rotations. However, their approach does only perform localization and no consistent reconstruction of the environment is created.

In our evaluation, we will compare a direct approach with a feature-based approach and point out their benefits and drawbacks.

To plan flying trajectories of an MAV, we need an accurate reconstruction of its environment. The process of generating this reconstruction is called mapping. A well established mapping framework is *OctoMap* [7] which uses range scans with known origins to model the world. A probabilistic occupancy estimation allows it to model occupied, free and also unknown areas. These areas are represented by a voxel based volumetric 3D model that is stored in an octree. *OctoMap* performs very well if it is fed with accurate dense range measurements. If these measurements are too sparse, many voxels will be labeled as unknown, which is due to the lack of interpolation. For obstacle avoidance this behavior is welcome because it denies any navigation through unknown space.

Our proposed approach contrasts from *OctoMap* in using sparse range measurements only. We compare both methods in terms of speed and accuracy using the EuRoC mapping evaluation framework.

In this paper, we describe our algorithms used in the European Robotics Challenge 3 Simulation Stage and especially evaluate the performance of the vision-based algorithms compared to others. As the challenge was divided into four tasks, we first describe and evaluate our solutions for the tasks localization and mapping. Then, we show how the results of the first two tasks could be used for state estimation, control and trajectory planning. This paper extends an extended abstract already submitted to the Austrian Robotics Workshop [13], where the focus was set more intensively on state estimation, control and trajectory planning.

2 Vision-Based Localization and Mapping

The first part of the simulation contest was split into the tasks of vision-based localization and mapping. A robust solution for both tasks is essential to achieve an MAV capable of safe autonomous navigation in GPS-denied environments.

2.1 Localization

In this task, the goal was to localize the MAV using stereo images and synchronized IMU data only. The implemented solution had to run on a low-end CPU (similar to a CPU onboard an MAV) in real-time. The results were evaluated on datasets with varying difficulty (see Fig. 1) in terms of computation speed and local accuracy. We compared two purely visual algorithms for this task, which are presented in this section.

Our first approach is a visual odometry system based on *libviso2* [6]. It is a keypoint-based approach which applies a combination of blob and corner detectors for keypoint extraction. First, feature points are detected. However, as the resulting quantity of points is high, non-maximum suppression is applied on the feature points and bucketing is used to spread them uniformly over the image domain. Next, quad matching is performed, where feature points of the current and previous stereo pair are matched in a loop between the four frames. A match is found if the loop is closed and the first and last feature in the matching loop are the same. Finally, pose estimation is done by using a RANSAC scheme for the selection of the feature points and by minimizing the reprojection error using Gauss-Newton optimization.

The second algorithm implements a dense direct approach to perform visual odometry and is compared against the sparse approach. In our implementation,

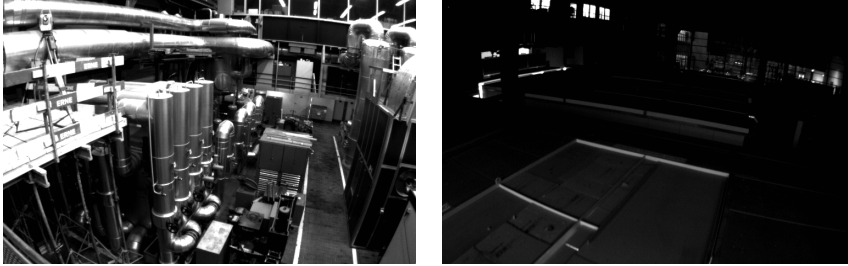


Figure 1: Input data for the localization task. *Left*: Image from the simple dataset. *Right*: Image from the difficult dataset. In comparison to the left image, the right image includes more poorly textured parts, over- and underexposed regions and more motion blur.

we compute a dense depth map for every keyframe using a fast depth map computation algorithm and estimate the pose of the frames between keyframes by minimizing the photometric error similar to [8]. To solve the minimization problem, the Levenberg-Marquardt algorithm is used. A new keyframe is created if the photometric error gets too big or if the rotational or translational movement from the previous keyframe to the current frame is exceeding a threshold.

2.2 Mapping

To successfully detect obstacles and circumnavigate them, an accurate reconstruction of the environment is needed. The goal of this task was to generate an occupancy grid of high accuracy in a limited time frame.

For our solution we only process frames from the stereo stream whose pose change to the previously selected keyframe exceeds a given threshold. From these keyframes we collect sparse features (approximately 100) that are extracted and matched using *libviso2* [6]. Using these features, we unproject 3D points and store them in a global point cloud with visibility information. After receiving the last frame, we put all stored data into a multi-view meshing algorithm based on [10]. It is an energy minimization based method that uses graph cuts. The meshing algorithm creates a Delaunay triangulation of the 3D points. A 3D Delaunay triangulation consists of vertices, facets and tetrahedra. A graph cut algorithm is applied which labels the tetrahedra of this triangulation as inside or outside. All facets that separate two tetrahedra with different labels compose the final surface, which we finally convert to an occupancy grid for evaluation. An example of our mapping process can be seen in Fig. 2.

We compare this approach with a second approach that uses the OctoMap [7] mapping framework. This framework needs to be fed with range scans from known origins, for which we use *libELAS* [5]. We have chosen *libELAS* because it is freely available and highly optimized to run fast on a single CPU. It also performs well on the KITTI Stereo-Evaluation benchmark, where it takes 48th place [1]. There it is one of the fastest methods that use CPUs only, thus it is well balanced in terms of speed and accuracy.

For both approaches we use the same set of keyframes and compare their results.

2.3 Results

For each task, the final scoring of the EuRoC was calculated using three different image datasets of varying difficulty and using the same computer for all the participants. Three test datasets of similar characteristics were provided to the contestants, which are used to perform the evaluations in this paper. The final scores of the EuRoC resulted in performance metrics that are comparable to

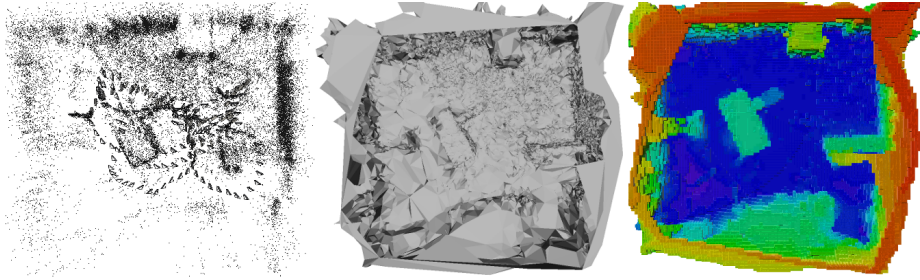


Figure 2: Mapping process. *Left*: 3D points and their keyframe camera poses. *Middle*: Constructed mesh. *Right*: Evaluated occupancy grid (scene height coded by color - from blue/low to red/high) .

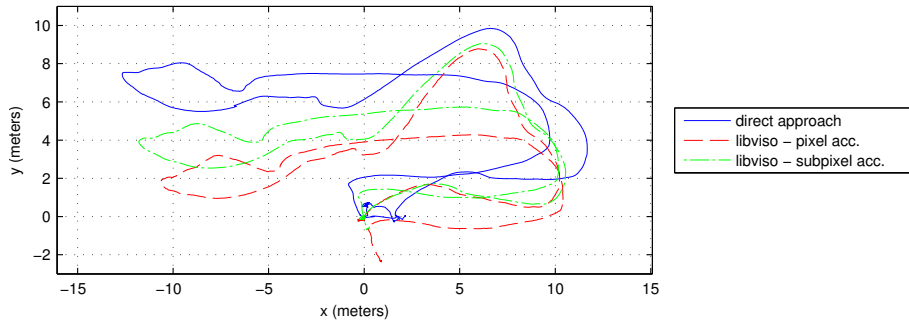


Figure 3: Trajectories in x- and y-direction (without height) of the evaluated approaches on Dataset 3 (difficult dataset). Our *direct* approach (blue) has an absolute translational error of 2.0781 m at the end of the trajectory, *libviso2* [6] with pixel accuracy (red) has an error of 2.4128 m and set to subpixel accuracy (green), it reaches an error of 1.1456 m.

performing the evaluation on these test datasets. The datasets contain stereo images with a resolution of 752x480 pixels each, with a baseline of 11 cm and a framerate of 20 Hz.

We performed our computations on a computer with a Quad Core i7-2630QM, @ 2.0 GHz. However, in accordance to the EuRoC, our solutions run in a virtual machine just using two cores of this processor.

For localization, we evaluate our two proposed algorithms in terms of speed and local accuracy on the test datasets. The three datasets have varying difficulty in terms of lightning conditions, scene depth, texture richness of the environment and motion blur (see Fig. 1). The difficulty increases from Dataset 1 to Dataset 3.

The sparse, feature-based approach is implemented using *libviso2* [6], a highly optimized visual odometry library. For the dense, *direct* approach, we use OpenCV’s block matching to compute a dense depth map and our own dense pose tracker similar to [8], which is also implemented efficiently using SSE instructions for most of the computational expensive tasks. Fig. 3 shows the computed trajectories of the used approaches.

The local accuracy is evaluated by computing the relative error as defined in the KITTI vision benchmark suite [4]. As we only have an exact ground truth for the 3D position and not for the rotation, only the translational error is evaluated. We measure the relative translational error of a trajectory distance of 2, 5, 10, 15, 20, 25, 60 and 75 meters and average these values to get a final translational error. The average runtime and translational errors for each dataset can be found in Table 1.

	Dataset 1		Dataset 2		Dataset 3	
	runtime	err_{trans}	runtime	err_{trans}	runtime	err_{trans}
<i>libviso2</i> - pixel	26 ms	1.8 %	26 ms	3 %	23 ms	6.2 %
<i>libviso2</i> - subpixel	34 ms	1.6 %	33 ms	1.4 %	30 ms	3.4 %
<i>direct</i>	49 ms	1.5 %	52 ms	2.2 %	52 ms	5.9 %

Table 1: Average runtime per frame and average translational error. *libviso2* [6] with subpixel accuracy outperforms our *direct* approach on Dataset 2 and 3. Even though our *direct* approach reaches a slightly higher accuracy on Dataset 1, the runtime is higher compared to *libviso2*.

	Relative translational error (in %)							
	2 m	5 m	10 m	15 m	20 m	25 m	60 m	75 m
<i>libviso2</i> - pixel	7.7	6.6	6.3	6.9	7.1	7.0	4.6	3.6
<i>libviso2</i> - subpixel	4.5	3.6	3.4	3.6	3.6	3.5	2.8	2.1
<i>direct</i>	11.8	7.6	5.9	5.6	5.7	5.6	2.9	2.4

Table 2: Relative translational error with varying trajectory distances for Dataset 3. The results of the *direct* approach contain more errors in small distances, while the accumulated error of *libviso2* [6] in longer distances is similar or higher.

The fastest approach is *libviso2* with pixel accuracy. It needs approximately half of the processing time of the dense approach but is slightly worse in terms of local accuracy on all datasets. However, using *libviso2* with subpixel accuracy is a good trade-off in terms of runtime and accuracy. It is still faster than the dense approach, and reaches a better accuracy in two of three datasets. Therefore, we chose to use *libviso2* in our final solution.

Table 2 illustrates the individual strengths and weaknesses of both approaches. Especially when using images with insufficient texture and/or containing motion blur (as in Dataset 3), jumps occur occasionally in the trajectory of the *direct* approach. Therefore, the error in small distances is bigger than compared to *libviso2*. Contrary, as *libviso2* estimates the relative pose from frame to frame and does not use keyframes, the accumulated error over bigger distances is similar or higher even though the error of small distances is much lower.

For the mapping task, the datasets contain a stereo stream and additionally the full 6-DoF poses captured by a Vicon system. The evaluation is done based on occupancy grids for three datasets (1 .. 3) of an indoor scene with increasing difficulties. At the lowest difficulty, the sensor moves slowly and smoothly and the room is well and homogeneously illuminated. The captured elements also have a good texture and the scene mainly consists of planar surfaces. With increasing difficulty, the motion of the sensor changes to a jerky up-and-down movement with a lot of rotational change only. In addition, the illumination changes frequently and the captured elements consist of fine parts that are challenging to reconstruct (e.g. a ladder). In all three datasets no moving objects are present. The ground truth occupancy grids were captured with a 3D laser scanner. The accuracy is evaluated by shifting a bounding box containing the MAV through the ground truth and the computed occupancy grids simultaneously. For each position of the bounding box a collision check in both occupancy grids is performed. This check can have four different states: correct-collision, missed-collision, correct-free and false-collision. The overall accuracy is calculated using the Matthews correlation coefficient (MCC). Its value can be between -1.0 and 1.0 , where 1.0 indicates that all collision checks

Dataset	Frames	Key-Frames	Accuracy[MCC]		Runtime[s]	
			OctoMap +libELAS	Ours	OctoMap +libELAS	Ours
1	2839	214	0.514	0.886	73	30
2	1559	284	0.589	0.896	82	23
3	2079	432	0.581	0.938	115	31

Table 3: Accuracy and runtime measurements of our approach and *OctoMap* [7]+*libELAS* [5].

	Dataset 1		Dataset 2		Dataset 3	
	OctoMap +libELAS	Ours	OctoMap +libELAS	Ours	OctoMap +libELAS	Ours
false-collision	27.57	5.78	23.76	4.67	24.17	2.75
missed-collision	0.00	0.11	0.00	0.67	0.00	0.39
correct-collision	54.13	54.02	51.84	51.17	52.30	51.90
correct-free	18.29	40.07	24.40	43.49	23.53	44.95

Table 4: Detailed collision check results of our approach and *OctoMap* [7]+*libELAS* [5]. All values are in percent[%] and rounded to two decimals. A high false or missed-collision rate decreases the MCC, correct recognition rates increase it.

have the correct state.

Table 3 shows the overall results. Our approach significantly outperforms *OctoMap* [7] +*libELAS* [5] in terms of speed. This is mainly caused by the runtime of the dense range measurement extraction. Although *libELAS* is fast, it needs 250 ms on average for a dense range measurements extraction. The sparse extraction performed by *libviso2* [6] for our approach needs 50 ms on average per key frame.

In terms of reconstruction accuracy, our approach outperforms *OctoMap* +*libELAS* too. If we look at the exact figures in Table 4, we see that *OctoMap*+*libELAS* always tends to be on the safe side. It never marks an occupied voxel as free, thus it never misses a collision. This circumstance is entirely desired for a secure navigation of an MAV through its environment. However, it marks many free voxels as occupied which could be used for path optimization. Our approach misses some collisions but it does not block as many feasible paths. This results in a significantly lower MCC score for all three datasets compared to our approach. The main advantage of our approach is a highly accurate map at a low computational cost.

3 State Estimation, Control and Navigation

The second track aimed at the development of a control framework to enable the MAV to navigate through the environment fast and safely. For this purpose, a simulation environment was provided by the EuRoC organizers where the hexacopter MAV dynamics were simulated in ROS/Gazebo.

The tasks' difficulty increased gradually from simple hovering to collision-free point-to-point navigation in a simulated industry environment. The evaluation included the performance under influence of constant wind, wind gusts as well as switching sensors.

3.1 State Estimation and Control

For state estimation, the available sensor data is a 6DoF pose estimate from an onboard virtual vision system (the data is provided at 10 Hz and with 100 ms delay), as well as IMU data (accelerations and angular velocities) at 100 Hz and with negligible delay, but slowly time-varying bias. Pose estimates returned

from task 1 could also be used here. However, for a comparable evaluation, synthetic data was used.

During flight, the position and orientation are tracked using a KALMAN-filter-like procedure based on a discretized version of [12]: the IMU sensor data are integrated using EULER discretization (*prediction* step); when an (outdated) pose information arrives, it is merged with an old pose estimate (*correction* step) and all interim IMU data is re-applied to obtain a current estimate. Orientation estimates are merged by turning partly around the relative rotation axis. The corresponding weights are established *a priori* as the steady-state solution of an Extended Kalman Filter simulation.

For control, a quasi-static feedback linearization controller with feedforward control similar to [3] was implemented. First, the vertical dynamics are used to parametrize the thrust; then, the planar dynamics are linearized using the torques as input. With this controller, the dynamics around a given trajectory in space can be stabilized via pole placement using linear state feedback; an additional PI-controller is necessary to compensate for external influences like wind.

The trajectory is calculated online and consists of a point list together with timing information. A quintic spline is fitted to this list to obtain smooth derivatives up to the fourth order, guaranteeing jerk and snap free trajectories.

3.2 Trajectory planning

Whenever a new goal position is received, a new path is delivered to the controller. In order to allow fast and safe navigation, the calculated path should stay away from obstacles and be smooth. Our approach is similar to the approach proposed by Richter et al. [14], where a trajectory is planned in 3D space based on a traditional planning algorithm and, then, used to fit a high order polynomial.

Our method proceeds as follows: First, the path that minimizes a cost function is planned, which penalizes proximity to obstacles, length and unnecessary changes in altitude. Limiting the cost increase, the raw output path from the planning algorithm is shortened. Finally, a speed plan is calculated based on the path curvature. The resulting path and timing information are used to fit the quintic spline used for feedforward control.

As input, we get a static map provided as an Octomap [7]. This map has similar properties as the map computed in task 2. However, in order to perform a comparable evaluation, synthetic data was used. To take advantage of the environment's staticity, we selected a Probabilistic Roadmap (PRM) based algorithm. As implementation, we used *PRMStar* from the *OMPL* library [15]. The roadmap and an obstacle proximity map are calculated prior to the mission. For the latter the *dynamicEDT3D* library [11] is used.

3.3 Results

The developed control framework achieves a position RMS error of 0.055 m and an angular velocity RMS error of 0.087 rad/s in stationary hovering. The simulated sensor uncertainties are typical of a multicopter such as the Asctec Firefly. The controlled MAV is able to reject constant and variable wind disturbances in less than four seconds.

Paths of 35 m are planned in 0.75 s and can be safely executed in 7.55 s to 8.8 s with average speeds of 4.2 m/s and peak speeds of 7.70 m/s.

4 Conclusions

We evaluated algorithms for vision-based real-time localization and mapping, which are suitable for low-end on-board computers of MAVs. Considering the

difficulty of the image datasets of the EuRoC challenge, these methods have proven their reliability to provide measurements to the proposed methods for state estimation, control and navigation. With this knowledge, we successfully participated in the EuRoC Simulation Contest. Future work will include deploying those algorithms onboard an MAV to achieve autonomous navigation.

ACKNOWLEDGMENT. We want to thank the Autonomous Systems Lab at ETH Zurich for providing the used data (including the ground truth). This project has partially been supported by the Austrian Science Fund (FWF) in the project V-MAV (I-1537), and a JAEPRe (CSIC) scholarship.

References

- [1] The kitti vision benchmark suite, 2015. http://www.cvlibs.net/datasets/kitti/eval_stereo_flow.php, 26.03.2015.
- [2] J. Engel, T. Schöps, and D. Cremers. Lsd-slam: Large-scale direct monocular slam. In *ECCV*, 2014.
- [3] O. Fritsch, P. D. Monte, M. Buhl, and B. Lohmann. Quasi-static feedback linearization for the translational dynamics of a quadrotor helicopter. In *American Control Conferenc*, June 2012.
- [4] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, 2012.
- [5] A. Geiger, M. Roser, and R. Urtasun. Efficient Large-Scale Stereo Matching. In *ACCV*, 2010.
- [6] A. Geiger, J. Ziegler, and C. Stiller. Stereoscan: Dense 3d reconstruction in real-time. In *IEEE Intelligent Vehicles Symposium*, 2011.
- [7] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard. Octomap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, 34(3):189–206, 2013.
- [8] C. Kerl, J. Sturm, and D. Cremers. Dense visual slam for rgb-d cameras. In *IROS*, 2013.
- [9] G. Klein and D. Murray. Parallel tracking and mapping for small ar workspaces. In *ISMAR*, 2007.
- [10] P. Labatut, J.-P. Pons, and R. Keriven. Efficient Multi-View Reconstruction of Large-Scale Scenes using Interest Points, Delaunay Triangulation and Graph Cuts. In *ICCV*, pages 1–8. IEEE, 2007.
- [11] B. Lau, C. Sprunk, and W. Burgard. Efficient grid-based spatial representations for robot navigation in dynamic environments. *Elsevier RAS2013*.
- [12] R. Mahony, T. Hamel, and J.-M. Pflimlin. Nonlinear complementary filters on the special orthogonal group. *IEEE TAC*, 2008.
- [13] J. Pestana, R. Prettenthaler, T. Holzmann, D. Muschick, C. Mostegel, F. Fraundorfer, and H. Bischof. Graz griffins’ solution to the european robotics challenges 2014. In *Austrian Robotics Workshop*, 2015.
- [14] C. Richter, A. Bry, and N. Roy. Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments. In *ISRR2013*.
- [15] I. A. Şucan, M. Moll, and L. E. Kavraki. The Open Motion Planning Library. *IEEE Robotics & Automation Magazine*, December 2012.